SYSTEMS AND METHODS FOR SOLVING NOGOOD DATABASES

BACKGROUND OF THE INVENTION

1. Field of Invention

[0001] This invention relates to systems and methods for solving nogood databases associated with a Boolean system of prepositional variables.

2. Description of Related Art

[0002] Constraint satisfaction problems involve the assignment of values to variables subject to a set of constraints. The set of constraints are an expression of knowledge about the variables and their relationships. Solving a constraint satisfaction problem involves finding a set of values for the variables that simultaneously satisfy the set of constraints. Constraint satisfaction problems include, for example, map coloring, understanding line drawings, electronic circuit analysis, and truth maintenance systems.

[0003] Many different approaches for solving constraint problems are known. For example, Freuder, "A Sufficient Condition for Backtrack-Bounded Search", Journal of the Association for Computing Machinery, Vol. 32, No. 4, pp. 755-761, October 1985, incorporated herein by reference in its entirety, discusses an approach to bounding the backtracking of a backtrack search. Dechter et al., "Network-based Heuristics for Constraint-Satisfaction Problems", Artificial Intelligence, 34, pp. 1-38, 1988, also incorporated herein by reference in its entirety, discusses a method of generating heuristic advise to guide the order of value assignments in solving constraint satisfaction problems. See also U.S. Patents 5,438,511, 5,727,222, 5,819,210, 5,903,860 and 6,064,953, each of which is incorporated herein by reference in its entirety.

SUMMARY OF THE INVENTION

[0004] A nogood database as used throughout the description of this invention is a database or a collection of data arranged for ease of retrieval, wherein the data comprises one or more nogoods. In the context of this invention, a nogood is a propositional variable or a conjunction of propositional variables whose associated constraints are unsatisfiable in the context of the current problem. If there is no valid

solution that contains the constraints associated with a propositional variable, the propositional variable is nogood. If there is no valid solution that contains all of the constraints associated with the propositional variables, the conjunction of propositional variables is nogood. A nogood database may comprise, for example, a collection of nogoods that are indexed according to the terms or variables that occur in the nogoods. In particular, a nogood database may comprise a representation of a constraint satisfaction problem.

[0005] This invention provides systems and methods that have improved efficiency for solving nogood databases.

[0006] This invention separately provides systems and methods for reducing the of nogoods from a list of contexted disjuctions.

[0007] This invention separately provides systems and methods for generating a nogood-free or backtrack-free list of contexted disjuctions.

[0008] This invention separately provides systems and methods that create a packed or efficient representation of all combinations of a set of variables subject to a set of constraints.

[0009] This invention separately provides systems and methods that can preserve independence of independent disjunctions.

[0010] In various exemplary embodiments of the systems and methods according to this invention, nogood databases are solved by generating a representation comprising a plurality of contexted disjunctions, conjoining all of the contexted disjunctions to form a conjunction of contexted disjunctions, and storing the representation as the conjunction of contexted disjunctions. In various embodiments, nogoods are reduced, and ideally eliminated, by refining the representation until a result of the conjunction of contexted disjunctions is more or less, or ideally completely, backtrack-free or the result of the conjunction of contexted disjunctions reduces to false. In various exemplary embodiments, refining the representation is carried out without reordering the disjunctions and/or without merging the disjunctions.

[0011] In other various exemplary embodiments of the systems and methods according to this invention, the representation is transformed so that the conjunction of contexted disjunctions is more or less, or ideally completely, backtrack-free. In various exemplary embodiments, transforming the representation is

carried out without reordering the disjunctions and/or without merging the disjunctions.

[0012] In other various exemplary embodiments of the systems and methods according to this invention, the representation is transformed so that choosing any disjunct from each of the disjunctions results in a valid solution. In various exemplary embodiments, transforming the representation is carried out without reordering the disjunctions and/or without merging the disjunctions.

[0013] In various exemplary embodiments of the systems according to this invention, a storage device stores a representation comprising a plurality of contexted disjunctions. A processor conjoins all of the contexted disjunctions to form a conjunction of contexted disjunctions and replaces the representation with the conjunction of contexted disjunctions. In various exemplary embodiments, the processor, or possibly another processor, reduces or ideally eliminates, nogoods by refining the representation until a result of the conjunction of contexted disjunctions is more or less, or ideally completely, backtrack-free or the result of the conjunction of contexted disjunctions reduces to false. In various other exemplary embodiments, the processor, or possibly another processor, transforms the representation so that the conjunction of contexted disjunctions is more or less, or ideally completely, backtrack-free. In still other various exemplary embodiments, the processor, or possibly another processor, transforms the representation so that choosing any disjunct from each of the disjunctions results in a valid solution.

[0014] These and other features and advantages of this invention are described in, or are apparent from, the following detailed description of various exemplary embodiments of the systems and methods according to this invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] Various exemplary embodiments of the systems and methods of this invention described in detail below, with reference to the attached drawing figures, in which:

[0016] Fig. 1 is a schematic representation of one exemplary embodiment of a system according to this invention;

[0017] Fig. 2 is an exemplary block diagram of the embodiment of Fig. 1; and

[0018] Fig. 3 is a flowchart illustrating one exemplary embodiment of a method for solving nogood databases according to this invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0019] This invention provides a new technique for solving nogood databases associated with a Boolean system of propositional variables. According to this invention, the Boolean system of propositional variables is represented as a conjunction of contexted disjunctions. This differs from representing the Boolean system of propositional variables as a disjunction of conjunctions, such as, for example, the disjunctive normal form (DNF), as is done conventionally.

[0020] According to this invention, the general form of the solution will be:

$$(A \vee B \vee \ldots) \wedge$$
$$(P \rightarrow (E \vee F \vee \ldots)) \wedge$$
$$(Q \rightarrow (I \vee K \vee \ldots)) \wedge$$
$$\ldots$$

where A, B, etc. are propositional variables and $X \rightarrow (Y \vee Z)$ represents a contexted disjunction where X is an arbitrary Boolean expression of propositional variables. A contexted disjunction is a disjunction that holds only in the given context (if X is true, then Y or Z is true). The choices in parentheses, such as $(E \vee F \vee \ldots)$, are mutually exclusive and do no occur in any of the disjuncts of the other disjunctions. The expression $(A \vee B \vee \ldots)$ is shorthand for $\text{TRUE} \rightarrow (A \vee B \vee \ldots)$.

[0021] According to this invention, the arbitrary Boolean expressions of propositional variables that appear as the contexts of disjunctions, such as X, are required to refer only to variables that appear before them, that is, in a higher-level conjunction. This allows solutions to be enumerated by: (1) making the first disjunction the current disjunction; (2) evaluating the context of the current disjunction; (3) if the context evaluates as TRUE, non-deterministically choosing a disjunct of the disjunction and setting the propositional variable of the chosen disjunct to TRUE; (4) making the next disjunct the current disjunction; and (5) repeating steps (2)-(4) until all disjunctions of the representation have been made the current disjunction.

[0022] The resulting enumeration is backtrack-free if there are no nogoods. In other words, if there are no nogoods, then every choice that can be made will lead to a valid solution.

[0023] Conventional techniques for solving nogood databases involve representing the Boolean system of propositional variables in disjunctive normal form (DNF), pruning simpler choices during the enumeration and filtering the result by more complex nogoods when the enumeration is complete. When many independent disjunctions exist, the resulting number of solutions may be exponential in the number of disjunctions. For example, if there are 30 disjunctions, there may be up to 2^{30} solutions.

[0024] According to this invention, all of the contexted disjunctions are conjoined together. Then, the result is "freed" relative to the nogoods. For example, in the following conjunction:

$$(P \rightarrow (P1 \vee P2)) \wedge$$
$$(P1 \rightarrow (Q1 \vee Q2)) \wedge$$
$$(Q2 \rightarrow (R1 \vee R2))$$

$(P1 \wedge R1)$ is a nogood.

[0025] The representation isn't "free" because disjuncts cannot be chosen freely from the alternatives while simultaneously guaranteeing a valid solution. For example, if P1, Q2 and R1 are chosen, then a nogood result is obtained. According to the invention, the representation is "freed" by refining or transforming the representation so that the disjuncts can be chosen freely from the alternatives while simultaneously guaranteeing a valid solution.

[0026] The disjunctions cannot be reordered or merged because doing so would violate the requirement that the variables in the context of a disjunction must be set by prior disjunctions, as noted above.

[0027] One way to "free" the representation is to refine or transform the representation to the following:

$$(P \rightarrow (P1 \vee P2)) \wedge$$
$$(P1 \rightarrow (Q1 \vee Q2)) \wedge$$
$$(Q2 \wedge P1 \rightarrow R2) \wedge$$
$$(Q2 \wedge \neg P1 \rightarrow (R1 \vee R2))$$

by splitting $Q2 \rightarrow (R1 \vee R2)$ into $Q2 \wedge P1 \rightarrow R2$ and $Q2 \wedge \neg P1 \rightarrow (R1 \vee R2)$ to isolate the situation where both P1 and R1 are true at the same time and pruning R1 where it occurs in the context of P1, since $P1 \wedge R1$ is a nogood.

If pruning R1 were to eliminate the last disjunct in the disjunction, then the context would become nogood and would be added to the nogood database to be processed.

[0028] The resulting representation is a conjunction of contexted disjunctions that is backtrack-free. If P1 $\wedge$ Q and Q2 $\wedge$ R are picked, then R1 cannot be picked because the only disjunction whose context is true is R $\wedge$ P1 $\rightarrow$ R2.

[0029] In general, the contexted disjunctions are "freed" one nogood at a time. For each nogood, a list of each disjunction is made such that (1) the disjunction mentions one of the nogood's propositional variables in its disjunctions, and (2) all of the other nogood variables occur in disjunctions before the current disjunction. There may be more than one such disjunction because of splits caused by previous nogoods.

[0030] Each disjunction is then split into two mutually exclusive disjunctions based on the nogood. This is done by eliminating the disjunction's variable from the nogood, and then conjoining the reduced nogood with the first disjunction and the negation of the reduced disjunction with the second disjunction. The nogood's variable is then pruned from the first disjunction's list of disjuncts. If the first disjunction becomes empty, then the context of the disjunction is made nogood, adding it to the nogood database in such a way that it will be processed with the rest of the nogoods.

[0031] The above approach assumes that nogoods are represented as simple conjunctions of propositional variables. If the nogoods are not in this form, but also include disjunctions and negations, then the nogoods can be converted to this form by using standard techniques involving disjunctive normal form (DNF) and DeMorgan's Law. Alternatively, one may create a truth table of the variables involved in the nogood and use the lines in the truth table that are true as input to the above approach.

[0032] The above approach may also be improved by processing the nogoods in the nogood database more intelligently. For example, one may simplify the nogoods and process the simplest nogoods first, or eliminate redundant nogoods.

[0033] Because of the approach provided by this invention, if the contexted disjunctions are mostly independent, then the number of disjunctions in the result will tend to be polynomial in the number of disjunctions in the input. This means that the output can be exponentially smaller than other techniques in some

circumstances. It also means that the above approach can produce the output in exponentially less time than other techniques in some circumstances.

[0034] Fig. 1 is a schematic representation of an exemplary embodiment of a system 100 for solving nogood databases according to this invention. According to this representation, a conjunction of contexted conjunctions 112 and associated nogoods 114 are processed to obtain a conjunction of backtrack-free contexted disjunctions 116.

[0035] Fig. 2 is an exemplary block diagram for one exemplary embodiment of the system shown in Fig. 1. As shown, the system 100 comprises a storage device 110 and a processor 120. The storage device 110 is used to store a representation comprising a plurality of contexted disjunctions, for example, the conjunction of contexted conjunctions 112. The storage device 110 is also used to store the associated nogoods 114. The processor 120 may be used to generate the conjunction of contexted conjunctions 112 and/or to identify the associated nogoods 114.

[0036] In various embodiments, the processor 120 processes the conjunction of contexted disjunctions 112 and refines the representation until a result of the conjunction of contexted disjunctions 112 is backtrack-free or the result of the conjunction of contexted disjunctions 112 reduces to false. In this manner, nogoods are removed from the conjunction of contexted disjunctions 112 to obtain the conjunction of backtrack-free contexted disjunctions 116. The processor 120 may refine the representation without reordering the disjunctions and/or without merging the disjunctions.

[0037] In other various embodiments, the processor 120 processes the conjunction of contexted disjunctions 112 and transforms the representation so that the conjunction of contexted disjunctions 112 is backtrack-free or so that choosing any disjunct from each of the disjunctions results in a valid solution. The conjunction of backtrack-free contexted disjunctions 116 is thus obtained. The processor 120 may transform the representation without reordering the disjunctions and/or without merging the disjunctions.

[0038] Fig. 3 is a flowchart illustrating one exemplary embodiment of a method for solving nogood databases according to this invention. Beginning in step S1000, operation continues to step S1010, where the first nogood is made the current

nogood. Then, in step S1020, a determination is made whether a current nogood exists. If not, operation jumps to S1210, where the process ends. Otherwise, operation continues to step S1030.

[0039] If a current nogood exists in step S1020, in step S1030, a list of relevant disjunctions is made. A disjunction is relevant if: (1) one of the nogood's variable is mentioned in the disjuncts; and (2) all of the other variables of the nogood occur in disjunctions before this disjunction. Next, in step S1040, the first disjunction is made the current disjunction. Then, in step S1050, a determination is made whether a current disjunction exists. If a current disjunction exists, operation continues to step S1060. Otherwise, operation returns to step S1020.

[0040] In step S1060, the current disjunction is split into mutually exclusive disjunctions based on the nogood. Then, in step S1070, nogood disjuncts are pruned from the current disjunction. Next, in step S1080, a determination is made whether the current disjunction is empty. If so, operation proceeds to step S1090. Otherwise, operation jumps directly to step S1100.

[0041] In step S1090, the context of the first disjunction is added to the nogood database, and continues to step S1100. Then, in step S1100, the next disjunction is made the current disjunction. Operation then returns to step S1050.

[0042] If a current disjunction does not exist, control jumps from step S1050 to step S1200, where the next nogood is made the current nogood, and returns to step S1020. If a current nogood does not exist, control jumps from step S1020 to step S1210, where the process ends.

[0043] The systems and methods for solving nogood databases according to this invention may be implemented on a programmed general purpose computer. However, the systems and methods for solving nogood databases according to this invention can also be implemented on a special purpose computer, a programmed microprocessor or micro-controller and peripheral integrated circuit elements, an ASIC or other integrated circuit, a digital signal processor, a hardwired electronic or logic circuit such as a discrete element circuit, a programmable logic device such as a PLD, PLA, FPGA or PAL, or the like. In general, any device, capable of implementing a finite state machine that is in turn capable of implementing the flowchart shown in Fig. 3 can be used to implement the systems and methods according to this invention.

[0044] The various blocks shown in Fig. 2 can be implemented as portions of a suitably programmed general-purpose computer. Alternatively, the various blocks shown in Fig. 2 can be implemented as physically distinct hardware circuits within an ASIC, or using a FPGA, a PDL, a PLA or a PAL, or using discrete logic elements or discrete circuit elements. The particular form each of the blocks shown in Fig. 2 will take is a design choice and will be obvious and predicable to those skilled in the art based on the above description.

[0045] In particular, the processor 120 shown in the exemplary embodiment may comprise any known or hereafter developed device and/or software that is capable of carrying out the various steps of generating a representation, conjoining contexted disjunctions, and refining and/or transforming a representation as described above. Similarly, the storage device 110 shown in the exemplary embodiment may comprise any known or hereafter developed device that allows storage and retrieval of information that constitutes a representation comprising a plurality of contexted disjunctions.

[0046] While this invention has been described in conjunction with various exemplary embodiments, it is to be understood that many alternatives, modifications and variations would be apparent to those skilled in the art. Accordingly, Applicant intends to embrace all such alternatives, modifications and variations that follow in the spirit and scope of this invention.

[0047] For example, it should be understood that the design and configuration of the system are illustrative and not limiting. The systems and methods of this invention may be used for various applications, either known or hereafter developed, that utilize the solution of constraint satisfaction problems. The particular design and configuration of the system may vary with the particular application.